Detecting the Higgs Boson Using Machine Learning Techniques

Filip Carević, Edvin Maid, Natalija Mitić
Machine Learning CS-433, EPFL, Switzerland

***Abstract*—Experiments in CERN are famous for producing great quantities of data, most of which is a product of unimportant events produced in the LHC. This fact renders experiments ideal for Machine Learning techniques. In this paper, we are proposing linear and logistic regression models for detecting the elusive Higgs Boson. We are also proposing methods for editing the data in order to make the training and inference more stable and successful. Finally, we demonstrate the potency of derivative-free optimization algorithms like random search and genetic algorithms for hyperparameter tuning.**

I. Introduction

The Higgs boson is a particle from the Standard Model in particle physics first hypothesized in the 1960s. Experiments in CERN have discovered a particle with the expected properties and generated vast collections. This paper proposes to leverage this data using regression techniques in order to determine whether the observed event, which produces 30 different features, is either a 'signal' (a Higgs boson) or 'background' (something other than a Higgs Boson).

II. Models and Methods

A. Data analysis and preprocessing

The dataset used for training, validation and testing, as well as the challenge, as described in [1], has a structure defined around a single feature (i.e. `PRI_jet_num`). To be precise, for different values of `PRI_jet_num` particular columns may be discarded as they have zero variance. Moreover, the data contains many entries with undefined (Not a Number) values, as well as outliers.

1) Meaningless Features: As mentioned before, some features are rendered meaningless for certain values of the categorical `PRI_jet_num` where $PRI\_jet\_num \in \{0, 1, 2, 3\}$. This distribution of meaningless features is presented in Table I. As it can be seen from the table, values 2 and 3 have the same meaningless features.

`PRI_jet_num`	Meaningless feature indexes
0	4,5,6,12,22,23,24,25,26,27,28,29
1	4,5,6,12,22,26,27,28
2 and 3	-

Table I
Features rendered meaningless for different values of PRI_JET_NUM

To address this, the problem can be split into three subsets based on the values of the `PRI_jet_num` feature. Consequently, three different models would be trained with different weights. Another way of addressing this is to convert the ordinal representation of the `PRI_jet_num` into a one-hot representation and put zero values for the meaningless features.

2) Imputation: After removing meaningless columns, the data still contains undefined values for the `DER_mass_MMC` column (15.24%). These values inhibit the normal functioning of linear regression models, hence, we need to replace these values with something meaningful. One imputation strategy was to use the mean value of the column extracted from the train set (not from the test set to avoid overfitting test data). The mean value strategy is, however, sensitive to outliers and as a consequence introduced instabilities during training. Another strategy that was used was to replace undefined values with the median, which is robust and produced considerably better results than the mean strategy.

3) Outliers: After a thorough analysis, we suspected that a lot of data may be considered as outliers. In order to remove those outliers, we applied two different methods: Tukey's fences method [2] and k-sigma rule [3]. Tukey's fences method considers a value to be an outlier if it is outside the range $[Q1 - k(Q3 - Q1), Q3 + k(Q3 - Q1)]$, while the k-sigma rule specifies valid range as $[mean - k \cdot std, mean + k \cdot std]$, for a given $k \in \mathbf{N}$. However, after the application of these methods, we discarded too much of the original data, which resulted in the model's underperforming 77% of the accuracy on the test data. In addition, similar results were achieved by clipping outliers to the mean or median values.

4) Splitting the data: The complete dataset used is comprised of 250'000 labeled data points. In order to successfully train the model, tune the hyperparameters and evaluate its performance on the unknown data, it is needed to split the data into three sections.

split	num of datapoints	percentage
train set	175000	70%
validation set	50000	20%
test set	25000	10%

Table II
The way the data has been split for training, validation and testing

B. Models

Different models and methods of optimization were used for this task. After dividing the problem into three sub-problems it was necessary to produce three models - each for a different domain of the dataset. Linear regression models were trained using Gradient Descent (GD), Stochastic Gradient Descent (SGD) and also Normal Equations (NE) for least squares and ridge regression. Additionally, logistic regression was trained with and without regularization using Gradient Descent

The presented models applied directly to the features are prone to under-fitting due to the model's simplicity. In order to achieve better performance (i.e. lower bias) we employed polynomial feature expansion. This is done on all features with different degrees of expansion. The way the degrees were assigned is explained in the next section II-C.

An important remark concerning logistic regression is the instability during training. To be precise, the output values that are produced by the model when calculating the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ for negative values, produces exponentially large outputs of the e^{-x} function, which exceeds minimum floating point values. To address this, we introduce a modifiable limit to the smallest value that the sigmoid function may output, and thus clipping the smallest negative input to a numerically viable threshold $\frac{1}{1+e^{-x}} > \epsilon \implies x > -\ln(\frac{1}{\epsilon} - 1)$.

C. Model selection and hyperparameter tuning

By introducing feature expansion the complexity of hyperparameters noticeably increases. It is therefore necessary to find a way to optimize the feature expansion. For this we have implemented two different methods 1) Random Search algorithm and 2) Genetic Algorithm [4]. The search algorithms are optimizing different degrees of expansion for each feature.

1) Random Search Algorithm: For all the parameters of a given model, the algorithm would generate an array of integers denoting the degree of expansion for each feature (if the degree was 0 the feature was removed altogether). Upon generating a new expansion it would compare its performance on the validation set with the $k = 10$ expansions with the highest model accuracy. In the end, the algorithm produces 10 different expansions which have the highest accuracy. This approach also varied other hyperparameters including the model type to find the best performing architecture.

2) Genetic algorithm: A typical genetic algorithm requires genetic representation of the solution domain and fitness function to evaluate the solution domain. Genetic representation was the array of degrees used in polynomial expansion, and the fitness function reflected the model's accuracy on the validation set. The selection phase preserved the best 20% of the population, the mutation phase incremented one randomly selected degree in the array, while the cross-over phase of the algorithm created new individuals by simply concatenating different portions of randomly selected ones. Since new models had to be trained for every new *degree-array*, execution of this algorithm required substantial hardware resources. Due to the lack of resources, algorithm parameters (e.g. size of population, number of generations) had to be drastically decreased, thus resulting in less than optimal performance of this hyperparameter tuning.

III. Results

After removing outliers, replacing undefined values, standardization and splitting the data into three groups with respect to the `PRI_jet_num` the performance results on the test set are shown on III

Model	Performance	
	Accuracy	F-score
Linear regression GD	0.688	0.630
Linear regression SGD	0.687	0.627
Least Squares NE	0.708	0.654
Ridge regression NE	0.703	0.645
Logistic regression GD	0.714	0.651
Logistic regression with regularization GD	0.714	0.651

Table III
PERFORMANCES OF MODELS WITH BASE FEATURES

To increase performance we apply polynomial feature expansion along with the random search and genetic algorithms. The results are displayed on IV

Algorithm	Best Model	Performance	
		Accuracy	F-score
Random Search	Least Squares NE	0.824	0.734
Genetic Algorithm	Least Squares NE	0.805	0.699

Table IV
PERFORMANCES OF MODELS WITH EXPANDED FEATURES AND DERIVATIVE-FREE OPTIMIZATION ALGORITHMS

IV. Discussion

A thorough analysis of the dataset revealed to us that the original data can be separated into independent subsets for each of which a regression model can be fitted. Moreover, the addition of derivative-free optimization for hyperparameter tuning and features expansion has produced a remarkable improvement over using non-augmented features. However, as said before, the high resource demand for these algorithms makes exploring large optimization populations hard.

V. Summary

In this paper we demonstrate the importance of understanding, analyzing and cleaning the data (i.e. imputing missing values) in order to produce a stable training set. Moreover, we illustrated the potency of leveraging derivative-free optimization algorithms for tuning hyperparameters.

References

[1] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kégl, and D. Rousseau, "The Higgs boson machine learning challenge," in *Proceedings of the NIPS 2014 Workshop on High-energy Physics and Machine Learning*, ser. Proceedings of Machine Learning Research, G. Cowan, C. Germain, I. Guyon, B. Kégl, and D. Rousseau, Eds., vol. 42. Montreal, Canada: PMLR, 13 Dec 2015, pp. 19–55. [Online]. Available: https://proceedings.mlr.press/v42/cowa14.html

[2] J. W. Tukey, *Exploratory Data Analysis*. Addison-Wesley, 1977.

[3] B. Everitt, *The Cambridge dictionary of statistics*. Cambridge, UK; New York: Cambridge University Press, 2002. [Online]. Available: http://www.worldcat.org/search?qt=worldcat_org_all&q=052181099X

[4] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, Jun. 1994. [Online]. Available: https://doi.org/10.1007/BF00175354